1 deterministic logic and arithmetic instructions by:

2 ensuing processing of Instruction #1 on the first cycle in which an OpCode, source

3 address and destination address are read from an Instruction Memory;

4 providing a source address to the interface blocks for Instruction #1 on the second

5 cycle, and ensuing processing of Instruction #2 in which the OpCode, source address and

6 destination address are read from the Instruction Memory;

7 when data for Instruction #1 is available from the interface blocks on the third cycle,

8 providing the source address to the interface blocks for Instruction #2, and ensuing processing

9 of Instruction #3 in which the OpCode, source address and destination address are read from

10 the Instruction Memory;

11 processing data from the interface blocks for Instruction #1 on the fourth cycle, and

12 when data for Instruction #2 is available from the interface blocks, providing the source

13 address to the interface blocks for Instruction #3 and ensuing processing of Instruction #4 in

14 which the OpCode, source address and destination address are read from the Instruction

15 Memory;

16 providing destination and write controls of Instruction #1 for the interface blocks on

17 the fifth cycle, processing data from the interface blocks for Instruction #2, and when data for

18 Instruction #3 is available from the interface blocks, providing the source address to the

19 interface blocks for Instruction #4 and ensuing processing of Instruction #5 in which the

20 OpCode, source address and destination address are read from the Instruction Memory; and

1   when Instruction #1 is retired on the sixth cycle, providing destination and write

2   controls of Instruction #2 for the interface blocks, processing the data from the interface

3   blocks for Instruction #3, and when data for Instruction #4 is available from the interface

4   blocks, providing the source address to the interface blocks for the instruction #5 and ensuing

5   processing of Instruction #6 in which the OpCode, source address and destination address are

6   read from the Instruction Memory.

1   22.   The host-fabric adapter as claimed in claim 21, wherein said Micro-Engine

2   (ME) is configured to ensure that data dependency between contiguous Micro-instructions is

3   dealt correctly.

1   23.   The host-fabric adapter as claimed in claim 1, wherein said Micro-Engine

2   (ME) processes multiple ME instructions in parallel, when said ME instructions are non-

3   deterministic instructions by:

4   ensuing processing of Instruction #1 on the first cycle in which an OpCode, source

5   address and destination address are read from an Instruction Memory;

6   providing the source address to the interface blocks for Instruction #1 on the second

7   cycle, and ensuing processing of Instruction #2 in which the OpCode, source address and

8   destination address are read from the Instruction Memory;

9   when data for Instruction #1 is available from the interface blocks on the third cycle,

10   and a conditional Jump instruction based on Flags is set for Instruction #2, ensuing

3

1   processing of Instruction #3 in which the OpCode, source address and destination address are

2   read from the Instruction Memory;

3   processing data from the interface blocks for Instruction #1 on the fourth cycle,

4   providing the source address to the interface blocks for Instruction #3, ensuing processing of

5   Instruction #4 in which the OpCode, source address and destination address are read from the

6   Instruction Memory;

7   when data for Instruction #3 is available from the interface blocks on the fifth cycle

8   if the Jump condition is not TRUE, ensuing processing of Instruction #5 in which the

9   OpCode, source address and destination address are read from the Instruction Memory; and

10   if the Jump condition is TRUE, ensuing processing of the conditional Jump

11 instruction in which the OpCode, source address and destination address are read from the

12 Instruction Memory corresponding to a Jump Address, and when Instruction #1 is retired on

13 the sixth cycle, flushing Instruction #3, providing the source address to the interface blocks

14 for the conditional Jump instruction corresponding to the Jump Address.

1   24.   The host-fabric adapter as claimed in claim 23, wherein said Micro-Engine

2   (ME) is configured to ensure that only latest data from the interface blocks is used and correct

3   data is written to the interface blocks.

1   25.   The host-fabric adapter as claimed in claim 1, wherein said Micro-Engine

2   (ME) processes multiple tasks in parallel by:

4

1   ensuing processing of Instruction #1 on the first cycle in which an OpCode, source

2   address and destination address are read from an Instruction Memory;

3   providing the source address to the interface blocks for Instruction #1 on the second

4   cycle, and ensuing processing of Instruction #2 indicating a Task Switching Instruction in

5   which the OpCode, source address and destination address are read from the Instruction

6   Memory;

7   when data for Instruction #1 is available from the interface blocks and there is no data

8   processing on the third cycle for Instruction # 2, ensuing processing of Instruction #3 for a

9   new task in which the OpCode, source address and destination address are read from the

10   Instruction Memory;

11   processing data for Instruction #1 from the interface blocks on the fourth cycle and

12   providing the source address to the interface blocks for Instruction #3 for the new task;

13   providing destination and write controls of Instruction #1 for the interface blocks on

14   the fifth cycle for the old task and, when data for the new task for Instruction #3 is available

15   from the interface blocks, providing the source address to the interface blocks for Instruction

16   #4 for a new task and ensuing processing of Instruction #5 for the new task in which the

17   OpCode, source address and destination address are read from the Instruction Memory;

18   when Instruction #1 is retired on the sixth cycle, processing data from the interface

19   blocks for Instruction #3 for the new task, and when data for Instruction #4 is available from

20   the interface blocks for the new task, providing the source address to the interface blocks for

21   Instruction #5 for a new task and ensuing processing of Instruction #6 for the new task in

5

1   which the OpCode, source address and destination address are read from the Instruction

2   Memory; and

3   when Instruction #2 is retired on the seventh cycle, providing destination and write

4   controls for the interface blocks for Instruction #3 for the new task, processing data from the

5   interface blocks for Instruction #4 for the new task, and when data for Instruction #5 is

6   available from the interface blocks for the new task, providing the source address to the

7   interface blocks for Instruction #6 for a new task and ensuing processing of Instruction #7 for

8   the new task in which the OpCode, source address and destination address are read from the

9   Instruction Memory.

1   26.   The host-fabric adapter as claimed in claim 1, wherein said Micro-Engine

2   (ME) is implemented to achieve a throughput of one instruction per clock for logic and

3   arithmetic instructions by processing multiple instructions in parallel in multiple pipelines.

1   27.   The host-fabric adapter as claimed in claim 1, wherein said Micro-Engine

2   (ME) is implemented to achieve a throughput of one instruction per clock for logic and

3   arithmetic instructions even in the event of data-dependency between contiguous instructions

1   28.   The host-fabric adapter as claimed in claim 1, wherein said Micro-Engine

2   (ME) is implemented to handle multiple instructions at any given time even in the event of

3   uncertainty of the next instruction to be executed

6

1   29.   The host-fabric adapter as claimed in claim 1, wherein said Micro-Engine

2   (ME) is implemented to achieve a throughput of one instruction per clock even in the case of

3   non-determinism of the next instruction to be executed.

1   30.   The host-fabric adapter as claimed in claim 1, wherein said Micro-Engine

2   (ME) is implemented to perform multi-tasking (multi-threading) with minimal hardware and

3   graceful integration into normal processing.

1   31.   The host-fabric adapter as claimed in claim 1, wherein said Micro-Engine

2   (ME) is implemented to perform multi-tasking (multi-threading) with non-duplication or non-

3   dedication of hardware computing resources per task.

1   32.   A host system, comprising:

2   a host processor;

3   a host memory;

4   a host fabric adapter connected to said host processor and said host memory via a

5   system bus, and installed to access to a switched fabric, said host fabric adapter comprising:

6   a host interface arranged to interface said system bus;

7   a serial interface arranged to receive and transmit data from said switched

8   fabric;

at least one Micro-Engine (ME) arranged to establish connections and support

data transfers, via a switched fabric, in response to work requests for data transfers;

interface blocks arranged to interface said switched fabric and said system bus,

and send/receive work requests and/or data for data transfers, via said switched fabric,

and configured to provide context information needed for said Micro-Engine (ME) to

process said work requests for data transfers, via said switched fabric,

wherein said Micro-Engine (ME) is implemented with a pipelined instruction

execution architecture to handle one or more ME instructions and/or one or more

tasks so as to process data for data transfers.

33.   The host system as claimed in claim 32, wherein said Micro-Engine (ME) is

implemented to achieve a throughput of one instruction per clock for logic and arithmetic

instructions by processing multiple instructions in parallel in multiple pipelines.

34.   The host system as claimed in claim 32, wherein said Micro-Engine (ME) is

implemented to achieve a throughput of one instruction per clock for logic and arithmetic

instructions even in the event of data-dependency between contiguous instructions

35.   The host system as claimed in claim 32, wherein said Micro-Engine (ME) is

implemented to handle multiple instructions at any given time even in the event of

uncertainty of the next instruction to be executed

36.   The host system as claimed in claim 32, wherein said Micro-Engine (ME) is implemented to achieve a throughput of one instruction per clock even in the case of non-determinism of the next instruction to be executed.

37.   The host system as claimed in claim 32, wherein said Micro-Engine (ME) is implemented to perform multi-tasking (multi-threading) with minimal hardware and graceful integration into normal processing.

38.   The host system as claimed in claim 32, wherein said Micro-Engine (ME) is implemented to perform multi-tasking (multi-threading) with non-duplication or non-dedication of hardware computing resources per task.

39.   The host system as claimed in claim 32, wherein said Micro-Engine (ME) processes multiple ME instructions in parallel, when said ME instructions are deterministic logic and arithmetic instructions by:

ensuing processing of Instruction #1 on the first cycle in which an OpCode, source address and destination address are read from an Instruction Memory;

providing a source address to the interface blocks for Instruction #1 on the second cycle, and ensuing processing of Instruction #2 in which the OpCode, source address and destination address are read from the Instruction Memory;

9

1   when data for Instruction #1 is available from the interface blocks on the third cycle,

2   providing the source address to the interface blocks for Instruction #2, and ensuing processing

3   of Instruction #3 in which the OpCode, source address and destination address are read from

4   the Instruction Memory;

5   processing data from the interface blocks for Instruction #1 on the fourth cycle, and

6   when data for Instruction #2 is available from the interface blocks, providing the source

7   address to the interface blocks for Instruction #3 and ensuing processing of Instruction #4 in

8   which the OpCode, source address and destination address are read from the Instruction

9   Memory;

10   providing destination and write controls of Instruction #1 for the interface blocks on

11   the fifth cycle, processing data from the interface blocks for Instruction #2, and when data for

12   Instruction #3 is available from the interface blocks, providing the source address to the

13   interface blocks for Instruction #4 and ensuing processing of Instruction #5 in which the

14   OpCode, source address and destination address are read from the Instruction Memory; and

15   when Instruction #1 is retired on the sixth cycle, providing destination and write

16   controls of Instruction #2 for the interface blocks, processing the data from the interface

17   blocks for Instruction #3, and when data for Instruction #4 is available from the interface

18   blocks, providing the source address to the interface blocks for the instruction #5 and ensuing

19   processing of Instruction #6 in which the OpCode, source address and destination address are

20   read from the Instruction Memory.

10

1 40.   The host system as claimed in claim 39, wherein said Micro-Engine (ME) is

2 configured to ensure that data dependency between contiguous Micro-instructions is dealt

3 correctly.

1 41.   The host system as claimed in claim 32, wherein said Micro-Engine (ME)

2 processes multiple ME instructions in parallel, when said ME instructions are non-

3 deterministic instructions by:

4   ensuing processing of Instruction #1 on the first cycle in which an OpCode, source

5 address and destination address are read from an Instruction Memory;

6   providing the source address to the interface blocks for Instruction #1 on the second

7 cycle, and ensuing processing of Instruction #2 in which the OpCode, source address and

8 destination address are read from the Instruction Memory;

9   when data for Instruction #1 is available from the interface blocks on the third cycle,

10 and a conditional Jump instruction based on Flags is set for Instruction #2, ensuing

11 processing of Instruction #3 in which the OpCode, source address and destination address are

12 read from the Instruction Memory;

13   processing data from the interface blocks for Instruction #1 on the fourth cycle,

14 providing the source address to the interface blocks for Instruction #3, ensuing processing of

15 Instruction #4 in which the OpCode, source address and destination address are read from the

16 Instruction Memory;

17   when data for Instruction #3 is available from the interface blocks on the fifth cycle

1   if the Jump condition is not TRUE, ensuing processing of Instruction #5 in which the

2   OpCode, source address and destination address are read from the Instruction Memory;

3   if the Jump condition is TRUE, ensuing processing of the conditional Jump

4   instruction in which the OpCode, source address and destination address are read from the

5   Instruction Memory corresponding to a Jump Address, and when Instruction #1 is retired on

6   the sixth cycle, flushing Instruction #3, providing the source address to the interface blocks

7   for the conditional Jump instruction corresponding to the Jump Address.

1   42.   The host system as claimed in claim 41, wherein said Micro-Engine (ME) is

2   configured to ensure that only latest data from the interface blocks is used and correct data is

3   written to the interface blocks.

1   43.   The host system as claimed in claim 32, wherein said Micro-Engine (ME)

2   processes multiple tasks in parallel by:

3   ensuing processing of Instruction #1 on the first cycle in which an OpCode, source

4   address and destination address are read from an Instruction Memory;

5   providing the source address to the interface blocks for Instruction #1 on the second

6   cycle, and ensuing processing of Instruction #2 indicating a Task Switching Instruction in

7   which the OpCode, source address and destination address are read from the Instruction

8   Memory;

1   when data for Instruction #1 is available from the interface blocks and there is no data

2   processing on the third cycle for Instruction # 2, ensuing processing of Instruction #3 for a

3   new task in which the OpCode, source address and destination address are read from the

4   Instruction Memory;

5   processing data for Instruction #1 from the interface blocks on the fourth cycle and

6   providing the source address to the interface blocks for Instruction #3 for the new task;

7   providing destination and write controls of Instruction #1 for the interface blocks on

8   the fifth cycle for the old task and, when data for the new task for Instruction #3 is available

9   from the interface blocks, providing the source address to the interface blocks for Instruction

10   #4 for a new task and ensuing processing of Instruction #5 for the new task in which the

11   OpCode, source address and destination address are read from the Instruction Memory;

12   when Instruction #1 is retired on the sixth cycle, processing data from the interface

13   blocks for Instruction #3 for the new task, and when data for Instruction #4 is available from

14   the interface blocks for the new task, providing the source address to the interface blocks for

15   Instruction #5 for the new task and ensuing processing of Instruction #6 for the new task in

16   which the OpCode, source address and destination address are read from the Instruction

17   Memory; and

18   when Instruction #2 is retired on the seventh cycle, providing destination and write

19   controls for the interface blocks for Instruction #3 for the new task, processing data from the

20   interface blocks for Instruction #4 for the new task, and when data for Instruction #5 is

21   available from the interface blocks for the new task, providing the source address to the

1   interface blocks for Instruction #6 for the new task and ensuing processing of Instruction #7

2   for the new task in which the OpCode, source address and destination address are read from

3   the Instruction Memory.